

Memory-Efficient Decoding of LDPC Codes

Jason Kwok-San Lee
University of California, Berkeley
Email: jlee@eecs.berkeley.edu

Jeremy Thorpe
Jet Propulsion Laboratory
California Institute of Technology
Email: thorpe@caltech.edu

Abstract—We present a low-complexity quantization scheme¹ for the implementation of regular (3,6) LDPC codes. The quantization parameters are optimized to maximize the mutual information between the source and the quantized messages. Using this non-uniform quantized belief propagation algorithm, we have simulated that an optimized 3-bit quantizer operates with 0.2dB implementation loss relative to a floating point decoder, and an optimized 4-bit quantizer operates less than 0.1dB quantization loss.

I. INTRODUCTION

Low-Density-Parity-Check (LDPC) codes[1] have recently received a lot of attention because of their excellent error-correcting capability. LDPC codes have been shown to be able to perform close to the Shannon limit[2]. In the past decade or so, much of the research on LDPC codes has focused on the analysis and improvement of codes under decoding algorithms with floating point precision. However, to make LDPC codes practical in the real world, the design of an efficient quantization scheme used in hardware implementation is crucial.

Belief propagation algorithm is used to decode LDPC codes. The standard belief propagation algorithm defines real-valued messages passing along edges in a code graph. The standard way to simulate this algorithm is to store and update the messages in a very accurate representation such as floating-point numbers. However, for very high-speed LDPC decoders, it is clear that the high complexity associated with computing and storing a very accurate representation is to be avoided if possible. Therefore, we propose a low-complexity LDPC quantization scheme to make efficient hardware implementation possible.

In this paper, we present a general quantization scheme whose parameters we have optimized to target regular (3,6) codes. In addition to being a test-bed for comparing quantized algorithms, this class of codes remains an appealing choice in rate $\frac{1}{2}$ applications that cannot tolerate the error floors typically induced in codes highly optimized to perform close to capacity.

II. MOTIVATION

The advantages of using a low-complexity quantization schemes are many. They include:

- 1) In the belief propagation algorithm, messages passing along edges in a code graph may have to be stored.

¹The work described was funded by the IND Technology Program and performed at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration.

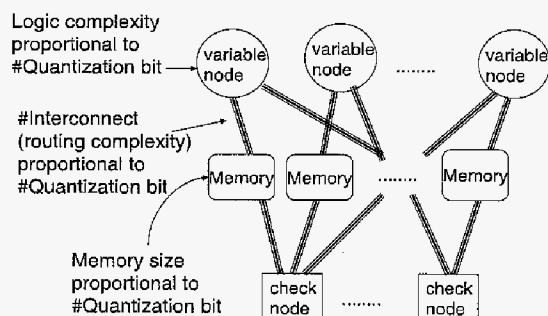


Fig. 1. Complexity proportional to quantization

The memory needed scales with the n -bit quantization as $O(n)$.

- 2) The number of interconnect wires to connect variable nodes and check nodes is proportional to the n -bit quantization. The complexity of interconnect routing scales at least linearly with n .
- 3) A smaller n -bit quantization makes it simpler for variable nodes and check nodes to update the messages. The logic complexity of variable nodes and check nodes units are often more than linear with n -bit quantization. In the worst case, an n -bit-input n -bit-output look-up table has logic complexity $O(2^n)$. Other schemes have complexity which scales as $O(n^2)$.

Recently, several research groups have developed LDPC decoders running on FPGA.[3][4] New generations of FPGA chips, such as Xilinx Virtex-II and Virtex-4, provide a sufficient amount of on-board block memory for the memory-demanding applications of digital signal processing. However, these devices also impose a practical constraint since the block memory is only divisible into 4-bit wide, or high-resolution such as 9-bit, 18-bit, or 36-bit.[5][6] Therefore, in order to utilize the on-board memory efficient, we should apply a n -bit quantization scheme compatible to the block memory division. 9-bit quantization provides very fine resolution, but can limit the size of code implementable in the device and can require significant amounts of power to be consumed. By comparison, an efficient 4-bit quantization can allow larger codes to be decoded, and is especially attractive if it can achieve small

quantization loss.

III. QUANTIZED BELIEF PROPAGATION ALGORITHM

In [8] a general non-uniform quantized belief propagation algorithm to decode regular LDPC codes is proposed. That scheme was a generalization of a message passing rule described in [9]. In it, the messages representing the likelihood ratios are essentially compressed by each computation node before being transmitted to the adjacent computation nodes.

The operation of each type of computation node (check and variable) occurs in a domain in which updates can be performed through simple additions and subtractions. For the variable nodes, this is essentially the log-likelihood-ratio (LLR) domain or "reliability" domain. For check nodes, the domain is called "unreliability" domain. Note that values in the two computational domains are typically represented by many more bits than are required to transmit and store inter-node messages.

The functions Q_v and Q_c which quantize the messages in the reliability domain and unreliability domain respectively into n -bit compressed messages. Complimentary to these are the functions ϕ_v and ϕ_c which restore the n -bit compressed messages into the computational domains of each node. Note that since variable nodes always send messages to check nodes and vice-versa, a message which is compressed from the reliability domain will always be restored into the unreliability domain, and vice-versa.

Initially, information from the channel is interpreted and quantized by a channel quantizer Q_{ch} which takes real-valued log-likelihood-ratios and produces a quantized representation. The function ϕ_{ch} takes a message produced by the channel quantizer and outputs a value to be used by the variable node.

At each iteration the variable node produces the messages $v_{i \rightarrow j}$. At iteration 0, the messages are given by $v_{i \rightarrow j}(0)$

$$v_{i \rightarrow j}(0) = Q_{ch}(\text{channel}_i), i \in \{1..n\} \quad (1)$$

At the t^{th} iteration, the parity check phase occurs first. All r check node units read the variable-to-check messages $v_{i \rightarrow j}$ from some edge memory connecting the i^{th} variable node to the j^{th} check node in the code graph, update the message by equation 2, then write the resulting check-to-variable messages $u_{j \rightarrow i}$ back to the edge memory according to the code graph connections.

$$u_{j \rightarrow i}(t) = Q_c(\sum_{i'} \phi_c(v_{i' \rightarrow j}(t-1))), j \in \{1..r\} \quad (2)$$

where i' ranges over all edges connected to the j^{th} check node excluding i , Q_c is the quantization rule for the check-to-variable message $u_{j \rightarrow i}$, and ϕ_c is the reconstruction function for the variable-to-check message $v_{i \rightarrow j}$. The architecture diagram of a check node unit is shown in Fig.2.

Next, the variable phase occurs. n variable node units read the check-to-variable messages $u_{j \rightarrow i}$ from edge memory,

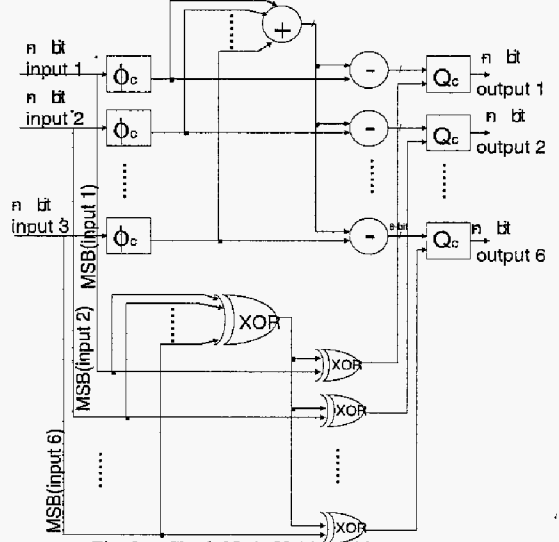


Fig. 2. Check Node Unit's architecture

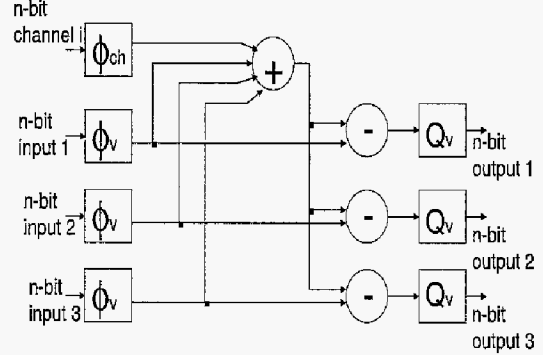


Fig. 3. Variable Node's architecture

update the message by equation 3, then write the variable-to-check messages $v_{i \rightarrow j}$ back to edge memory according to the code graph connections.

$$v_{i \rightarrow j}(t) = Q_v \left(\phi_{ch}(Q_{ch}(\text{channel}_i)) + \sum_{j' \neq j} \phi_v(u_{j' \rightarrow i}(t)) \right) \quad (3)$$

$, i \in \{1..n\}$

where j' ranges over all edges connected to the i^{th} variable node excluding j , Q_v is the quantization rule for the variable-to-check message $v_{i \rightarrow j}$, ϕ_v is the reconstruction function for the check-to-variable message $u_{j \rightarrow i}$, and ϕ_{ch} is the reconstruction function for the channel message $Q_{ch}(\text{channel}_i)$. The architecture diagram of a variable node unit is shown in Fig.3.

At the final K^{th} iteration, hard decisions X_i are made in variable nodes following:

$$X_i = \begin{cases} 0, & \sum_j u_{j \rightarrow i}(K) \geq 0 \\ 1, & \sum_j u_{j \rightarrow i}(K) < 0 \end{cases} \quad (4)$$

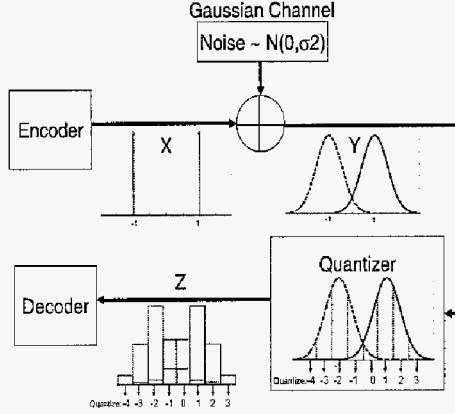


Fig. 4. Quantization of channel messages

IV. OPTIMIZING THE QUANTIZATION PARAMETERS

We targeted regular (3,6) codes to optimize the quantization parameters. In order to optimize the error-correcting performance in the quantization scheme, the intuition is to maximize the mutual information between the source and the quantized message. As the binary source signal is corrupted by the gaussian noise channel, the signal before the quantization process is a Gaussian-distributed real-valued message $Y = X + \text{noise}$. Therefore, the mutual information between X (binary source) and Y (Gaussian channel output) is:

$$I(X; Y) = 1 + \int_{-\infty}^{\infty} \Pr(y|x) \log_2 \left(1 + \frac{\Pr(y|x)}{\Pr(y)} \right) dy \quad (5)$$

Next, the n -bit quantizer maps the real-valued message Y into the appropriated quantized message Z according to the quantization parameters, $Z = Q_{ch}(Y)$. (See Fig.4) Therefore, the mutual information between the binary source X and the quantized message Z is:

$$I(X; Z) = 1 + \sum_z \Pr(z|x) \log_2 \left(1 + \frac{\Pr(z|x)}{\Pr(z)} \right) \quad (6)$$

In order to maximize the mutual information, the quantization function Q_{ch} is found such that:

$$\arg \max I(X; Z) = \arg \max \left\{ 1 + \sum_z \Pr(z|x) \log_2 \left(1 + \frac{\Pr(z|x)}{\Pr(z)} \right) \right\} \quad (7)$$

where

$$\Pr(z|x) = \int_{\min Q_{ch}^{-1}(z)}^{\max Q_{ch}^{-1}(z)} \Pr(y|x) dy \quad (8)$$

Once Q_{ch} is determined, initial values for ϕ_{ch} can be found by taking the midpoints of Q_{ch} quantized to an appropriate

TABLE I
OPTIMIZED 3-BIT QUANTIZATION PARAMETERS: RECONSTRUCTION
VALUES

x	$\phi_{ch}(x)$	$\phi_v(x)$	$\phi_c(x)$
-4	-21	-20	-1
-3	-15	-12	-2
-2	-9	-6	-6
-1	-3	-2	-26
0	3	2	26
1	9	6	6
2	15	12	2
3	21	20	1

TABLE II
OPTIMIZED 3-BIT QUANTIZATION PARAMETERS: QUANTIZER INTERVALS

$Q_{ch}(ch)/$ $Q_v(v)/$ $Q_c(c)$	ch	v	c
-4	$ch < -3.3$	$v < -18$	$-5 \leq c < 0$
-3	$-3.3 \leq ch < -2.2$	$-18 \leq v < -12$	$-9 \leq c < -5$
-2	$-2.2 \leq ch < -1.1$	$-12 \leq v < -6$	$-26 \leq c < -9$
-1	$-1.1 \leq ch < 0$	$-6 \leq v < 0$	$c < -26$
0	$0 \leq ch \leq 1.1$	$0 \leq v \leq 6$	$c > 26$
1	$1.1 < ch \leq 2.2$	$6 < v \leq 12$	$9 < c \leq 26$
2	$2.2 < ch \leq 3.3$	$12 < v \leq 18$	$5 < c \leq 9$
3	$ch > 3.3$	$v > 18$	$0 \leq c \leq 5$

scale. Initial values for the other four parameters can be found similarly.

After initial values of the quantization parameters are determined, these values are optimized using both simulation and density evolution. Currently, significant amount of hand-optimization is used, and we have not had time to explicate our optimization procedures in detail.

Using this strategy, we found several sets of optimized non-uniform quantization parameters, and listed as follows:

Table I: Optimized 3-bit Quantization rules: Reconstruction functions $\phi_{ch}(x)$, $\phi_v(x)$, and $\phi_c(x)$.

Table II: Optimized 3-bit Quantization rules: Quantizers' interval values $Q_{ch}(ch)$, $Q_v(v)$, and $Q_c(c)$.

Table III: Optimized 4-bit Quantization rules: Reconstruction functions $\phi_{ch}(x)$, $\phi_v(x)$, and $\phi_c(x)$.

Table IV: Optimized 4-bit Quantization rules: Quantizers' interval values $Q_{ch}(ch)$, $Q_v(v)$, and $Q_c(c)$.

V. SIMULATION PERFORMANCE

Using the optimized 3-bit and 4-bit quantization parameters targeted for regular (3,6) codes, we simulated our proposed non-uniform quantization scheme on a (4096, 2048) regular code. Using the 3-bit optimized quantizer, the LDPC decoder operates with 0.2dB implementation loss relative to a floating

6 is a known equation. Original argument of paper - need to maximize equation 6 in order to optimize quantization of LDPC code. This is using known code's quantization, just applying quantization technique to the code. Not yet in use - could be applied in future.

TABLE III
OPTIMIZED 4-BIT QUANTIZATION PARAMETERS: RECONSTRUCTION
VALUES

x	$\phi_{ch}(x)$	$\phi_v(x)$	$\phi_c(x)$
-8	-114	-114	-1
-7	-87	-87	-4
-6	-64	-64	-12
-5	-48	-48	-27
-4	-36	-36	-50
-3	-25	-25	-88
-2	-15	-15	-153
-1	-5	-5	-312
0	5	5	312
1	15	15	153
2	25	25	88
3	36	36	50
4	48	48	27
5	64	64	12
6	87	87	4
7	114	114	1

TABLE IV
OPTIMIZED 4-BIT QUANTIZATION PARAMETERS: QUANTIZER INTERVALS

$Q_{ch}(ch)$ or $Q_v(v)$ or $Q_c(c)$	ch	v	c
-8	$ch < 5.0$	$v < -210$	$-2 \leq c < 0$
-7	$-5.0 \leq ch < -3.7$	$-210 \leq v < -115$	$-7 \leq c < -2$
-6	$-3.7 \leq ch < -2.8$	$-115 \leq v < -67$	$-18 \leq c < -7$
-5	$-2.8 \leq ch < -2.1$	$-67 \leq v < -36$	$-36 \leq c < -18$
-4	$-2.1 \leq ch < -1.5$	$-36 \leq v < -18$	$-67 \leq c < -36$
-3	$-1.5 \leq ch < -1.0$	$-18 \leq v < -7$	$-115 \leq c < -67$
-2	$-1.0 \leq ch < -0.5$	$-7 \leq v < -2$	$-210 \leq c < -115$
-1	$-0.5 \leq ch < 0$	$-2 \leq v < 0$	$c < -210$
0	$0 \leq ch \leq 0.5$	$0 \leq v \leq 10$	$c > 210$
1	$0.5 < ch \leq 1.0$	$10 < v \leq 20$	$115 < c \leq 210$
2	$1.0 < ch \leq 1.5$	$20 < v \leq 30$	$67 < c \leq 115$
3	$1.5 < ch \leq 2.1$	$30 < v \leq 42$	$36 < c \leq 67$
4	$2.1 < ch \leq 2.8$	$42 < v \leq 56$	$18 < c \leq 36$
5	$2.8 < ch \leq 3.7$	$56 < v \leq 74$	$7 < c \leq 18$
6	$3.7 < ch \leq 5.0$	$74 < v \leq 100$	$2 < c \leq 7$
7	$ch > 5.0$	$v > 100$	$0 \leq c \leq 2$

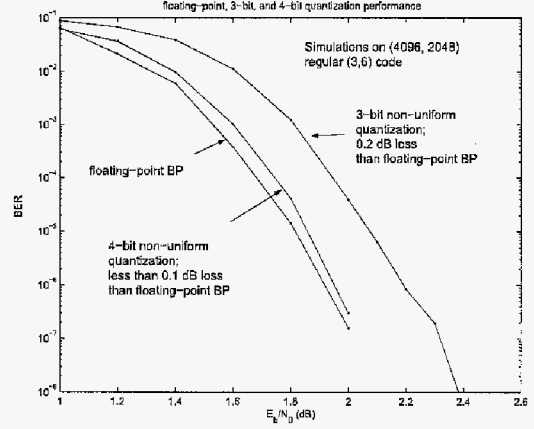


Fig. 5. Simulation performance of various quantization scheme

point belief propagation decoder. Using the 4-bit optimized quantizer, the LDPC decoder achieves quantization loss less than $0.1dB$. (See Fig. 5) For the sake of quantization simplicity, we adopted the optimized 3-bit quantizer into our FPGA-based structured LDPC decoder.[3]

VI. COMPARING NON-UNIFORM QUANTIZATION AND UNIFORM QUANTIZATION

[7] examined various uniform quantization schemes including uniform quantized offset BP-based decoding algorithms in details. While computationally more involved, our proposed non-uniform quantization schemes outperforms the uniform quantized counterpart when constrained by stored bit width. For example, decoding a regular (8000, 4000) LDPC code, [7]'s 5-bit uniform quantized offset BP-based algorithms suffers a degradation of $0.1dB$ compared with the unquantized BP algorithms. In comparison, simulating on a similar block-length (8192, 4096) regular LDPC code, our proposed 4-bit non-uniform quantization scheme operates less than $0.1dB$ implementation loss relative to a unquantized BP decoder. (See Fig. 6) Benefiting from a smaller quantization bit number while enjoying less implementation loss, non-uniform quantization may be preferable to be adopted in hardware implementation of LDPC decoder, especially on a FPGA-platform in which 4-bit quantization optimizes the block memory utilization.

VII. CONCLUSION

We have presented a general non-uniform low-complexity quantization scheme for the implementation of LDPC decoders, and demonstrated the 3-bit and 4-bit optimized quantization rules for regular (3,6) LDPC decoders. Maximizing the mutual information between the binary source and received quantized message allows the optimization of quantized LDPC decoding. As demonstrated by this work, an efficient low-complexity quantization can reduce the memory requirements and routing complexity in the hardware implementation of practical LDPC decoders.

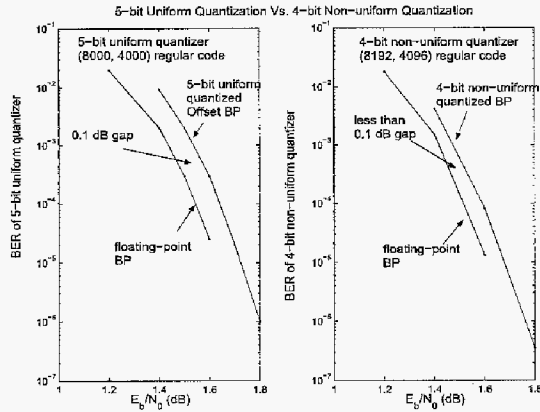


Fig. 6. 5-bit uniform quantizer Vs. 4-bit non-uniform quantizer

REFERENCES

- [1] R. G. Gallager, Low-Density Parity-Check Codes, MIT Press, Cambridge, MA, 1963.
- [2] S. Chung, G. D. Forney, T. J. Richardson, and R. Urbanke, "On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit," *IEEE Comm. Letters*, vol. 5, pp. 58-60, Feb. 2001.
- [3] J. Lee, B. Lee, J. Thorpe, K. Andrews, S. Dolinar, J. Hamkins, "A Scalable Architecture of a Structured LDPC Decoder," *Proc. IEEE ISIT 2004*, Chicago, Jun 27- Jul 2 2004, pp. 292.
- [4] T. Zhong and K. Parhi, "A 54 Mbps (3, 6)-Regular FPGA LDPC Decoder," *Proc. IEEE SIPS 2002*, San Diego, CA, Oct. 16-18, 2002, pp. 127-32.
- [5] 18 Kbit Block SelectRAM Resources, Xilinx Virtex-II Platform FPGAs Complete Data Sheet, pp. 21, <http://direct.xilinx.com/bvdocs/publications/ds031.pdf>
- [6] Block RAM Summary, Xilinx Virtex-4 User Guide, pp. 109, <http://direct.xilinx.com/bvdocs/userguides/ug070.pdf>
- [7] J. Chen, A. Dholakia, E. Eleftheriou, M. Fossorier, and X.-Y. Hu, "Reduced-Complexity Decoding of LDPC Codes."
- [8] J. Thorpe, "Low-complexity approximations to belief propagation for LDPC codes," available at <http://www.systems.caltech.edu/jeremy/research/papers/research.html>.
- [9] T. J. Richardson and R. L. Urbanke, "The Capacity of Low-Density Parity Check Codes Under Message-Passing Decoding," *IEEE Trans. on Info. Th.*, vol. 47, no. 2, pp. 599-618, 2001.